

Optimization of agricultural product storage using real-coded genetic algorithm based on sub-population determination

Wayan Firdaus Mahmudy¹, Nindynar Rikatsih^{1,2}, Syafrial³

¹Department of Informatics Engineering, Faculty of Computer Science, Universitas Brawijaya, Malang, Indonesia

²Department of Informatics, Institute of Technology, Science and Health of dr. Soepraoen Hospital, Malang, Indonesia

³Department of Agricultural Socio-Economic, Faculty of Agriculture, Universitas Brawijaya, Malang, Indonesia

Article Info

Article history:

Received Sep 13, 2021

Revised Mar 17, 2022

Accepted Apr 15, 2022

Keywords:

Agricultural product

Genetic algorithm

Knapsack problem

Migration

Sub-population

ABSTRACT

The storage of fresh agricultural products is a combinatorial problem that should be solved to maximize number of items in the storage and also maximize the total profit without exceed the capacity of storage. The problem can be addressed as a knapsack problem that can be classified as NP-hard problem. We propose a genetic algorithm (GA) based on sub-population determination to address the problem. Sub-population GA can naturally divide the population into a set of sub-population with certain mechanism in order to obtain a better result. GA based on sub-population is applied by generating a set of sub-population which is happened in the process of initializing population. A special migration mechanism is developed to maintain population diversity. The experiment shows GA based on sub-population determination provide better results comparable to those achieved by classical GA.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Wayan Firdaus Mahmudy

Department of Informatics Engineering, Faculty of Computer Science, Brawijaya University

Ketawanggede, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145, Indonesia

Email: wayanfm@ub.ac.id

1. INTRODUCTION

Fresh agricultural products are a daily need for the community. The products are perishable and must be stored at the proper condition before being shipped to distributors or retailers [1]. In doing product storage, we need to pay attention to the product value and the profit that must be gained. For agricultural products that the stock needs to be always available with a good quality, trader has to maximize number of items in the storage and maximize the total profit without exceed the capacity of storage. Therefore, trader should do the right way of product storage to get maximum profit. The storage of fresh agricultural products problem could be addressed as a knapsack problem.

Knapsack problem is a combinatorial and optimization problem that is often encountered in daily life and real industrial problems such as project selection, capital budgeting, cargo loading, and bin packing. [2], [3]. Knapsack problem also has a set of application for budgeting project, selection of items, material, and cost-effective development [4]–[6]. Optimization of knapsack problems is implemented to determine a number of items with a certain value that will be included into a container without exceeding the capacity of the container. The item selection is expected to provide maximum profit [7].

Various methods have been developed to solve knapsack problems such as local search, heuristics, meta-heuristic, and hybridization methods. For example, a local search is modified to efficiently solve a large scale knapsack problem [8]. An evolutionary algorithm based approach is developed to solve hardware/software partitioning that is considered as a variant of knapsack problem [9]. A hybrid approaches are also developed to address the complexity of the knapsack problem. For instance, ant colony optimization

and differential evolution algorithm are combined to explore strong characteristic of each method to solve different part the knapsack problem [10].

In this research we propose genetic algorithm (GA) to solve the problem because it is simple, easy to use and has a wide search area. GA is one of the meta-heuristic methods which has been proven that can be used to solve knapsack problems [11], [12]. GA is a meta-heuristic search algorithm that can provide optimal solutions by adopting mechanism of biological evolution and natural selection [13], [14]. GA consists of several steps including population initialization, crossover and mutation as a reproductive process to produce new solutions and the last is selection to get the best solution [15], [16]. GA is proved to effectively solve the optimization problem [17]–[19]. However, GA has weakness that is easy to be trapped in local optimum [20], [21]. It happens when the population of GA reaches a suboptimal state that the genetic operators produce offspring with a performance that can not be better than their parents [22]. The previous research uses dynamic genetic clustering algorithm and elitist technique to prevent premature convergence in GA [9]. A proper combination of crossover and mutation methods may be used to increase the GA performance [23]. The other research uses hybrid adaptive GA to overcome GA weakness. It combines GA with other algorithms to get better solutions [24]–[28]. The other way to solve GA weakness is by giving random injection [29].

To solve that weakness of GA, we propose GA based on subpopulation. This approach is adopting mechanism of parallel GA with the population that is naturally divided into a number of sub-populations that evolve and converge with a significant independence level [30]. Parallel GA can improve computational efficiency over classical GA. It also facilitates parallel exploration of solution space to get the better solutions [31]. GA based on sub-population determination solves GA weakness by keeping individual variety by setting the best individual of one sub-population to another population. A special migration mechanism is developed to maintain population diversity [32]. It is applied to get out of local optimum that cause premature convergence and increase the quality of the solutions. Therefore, in this research we propose GA based on sub-population determination to solve knapsack problem of agricultural product storage.

2. THE PROPOSED APPROACH

This section is divided into three parts that consists of problem statement, proposed fitness function to evaluate the quality of solutions, and mechanism of sub-population GA (SPGA) with a special migration. First, we make the detail of statement problem to prepare the basic formulation to create fitness function. Then, we present fitness function before entering the main procedure of SPGA.

2.1. Problem statement

The problem addressed in this study is optimization of agricultural product storage that is conducted by traders. The agricultural product is stored in storage with certain quantity in kilograms without exceed the capacity and represented by product 1, product 2, product 3, and so on. In this study, it is assumed that the available capacity is 5,000 kg. Although storing more items will potentially get higher profits, but there are additional costs that must be incurred if the items exceed the storage capacity. If the agricultural product quantity exceeds the available capacity, it is necessary to rent a storage place to another party at a cost of IDR 100 per kg. Example of possible solutions are presented in Table 1.

Table 1. Example of product quantity combination

No	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6	Product 7	Product 8
1	4,000	1,500	2,000	200	200	150	150	150
2	5,500	4,000	1,000	300	500	150	100	200
3	3,000	5,000	1,500	200	100	150	300	200
4	5,000	2,000	1,500	300	170	100	250	150
5	2,000	4,000	3,000	100	100	250	150	100

Based on Table 1 there are five combinations of the product quantity where the first one shows that the quantity of product 1 is 4,000 kg, product 2 is 1,500 kg, product 3 is 2,000 kg and the last product which is product 8 is 150 kg. Trader should find the appropriate quantity combination of the product because the quantity combination affects the profit. The calculation of gaining profit is showed in (1) to (5).

$$TC = (\sum_{i=1}^n Px_{1i}Q_i) + Px_2 \quad (1)$$

$$TR = (\sum_{i=1}^n Py_{1i}Q_i) \quad (2)$$

$$TR_{current} = \sum_{i=1}^n (Py_{1i}Qy_{1i} + Py_{2i}Qy_{2i}) \quad (3)$$

$$TPF_{max} = TR - TC \quad (4)$$

$$Potential\ losses = TR_{current} - TC \quad (5)$$

TPF is the total profit which is gained from total revenue and total cost. Total revenue is represented by TR while total cost is represented by TC . TC is obtained from the purchase price per product (Px_1) multiplied by quantity (Q) and summed by other expenses represented by Px_2 . TR is the selling price per product represented by Py_1 and multiplied by Q . Maximum profit represented by TPF_{max} . TPF_{max} is obtained by all products that are stored as many as Q without exceeding the capacity. Those all products also should be sold out. However, not all products are sold out because the products are sold according to demand represented by Qy_1 with a selling price of Py_1 so that there was a remaining stock of Qy_2 which experienced a decrease in the selling price of Py_2 . Therefore, the actual income obtained is the sales profit based on demand and stock sales with a decrease in price which is defined as the current total revenue $TR_{current}$. So that the profit is generated from total income minus expenses, namely as potential losses. An example of calculating profit is presented in Table 2.

Table 2. Profit calculation

Product	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6	Product 7	Product 8
Product quantity (kg)	4,000	1,500	2,000	200	200	150	150	150
Purchase cost per kilogram (Rp)	3,200	5,500	1,500	3,000	8,000	9,000	5,000	2,500
Purchase cost of each product (Rp)	12,800,000	8,250,000	3,000,000	600,000	1,600,000	1,350,000	750,000	375,000
Total purchase cost (Rp)				28,725,000				
Expenses (Rp)				3,075,000				
Excess capacity cost (Rp)				335,000				
Total cost (TC)				32,135,000				
Market demand (kg)	3,000	1,000	1,500	160	180	100	120	100
Selling price per kilogram	3,700	6,500	2,000	5,000	9,000	11,000	6,000	4,000
Selling price according to market demand	11,100,000	6,500,000	3,000,000	800,000	1,620,000	1,100,000	720,000	400,000
Total revenue (TR)				25,240,000				
Stock (kg)	1,000	500	500	40	20	50	30	50
Selling price of stock per kilogram	3,500	6,000	1,000	4,500	8,500	10,000	5,000	3,000
Selling price of stock	3,500,000	3,000,000	500,000	180,000	170,000	500,000	150,000	150,000
Total revenue (TR) of stock				8,150,000				
Current total revenue (TR current)				33,390,000				
Maximum selling price	14,800,000	9,750,000	4,000,000	1,000,000	1,800,000	1,650,000	900,000	600,000
Maximum total revenue max (TR max)				34,500,000				
Maximum total profit (TPF max)				2,365,000				
Potential Losses				1,255,000				

2.2. Proposed fitness function

Fitness value represents the quality of the solution produced by GA. The solution that provides maximum benefit is considered as a good solution. High fitness value represents high quality of solution which shows the high profit that can be provided. The determination of fitness function depends on calculating profit based on formulas (1) to (5). Hence, we consider the fitness function that is showed in (6).

$$fitness = \frac{1000}{TPF_{max} - potential\ losses} \quad (6)$$

2.3. Special migration on sub-population genetic algorithm (GA)

The GA is proposed by John Holland in 1975. GA is a heuristic method that mimic the mechanism of biological evolution and applies natural selection to obtain optimal solutions [5]. We apply classical GA with sub-population to solve the problem in this study.

Sub-population GA which is called SPGA in this research adopts mechanism of parallel GA with the population that is naturally divided into a number of sub-populations [33]. Similarity of sub-population GA and single-population GA is both of them applies several steps that are preceded by initialization of the population containing chromosomes that represent the solution. Furthermore, chromosomes are developed to get new variations by using crossover mutation process and selection. However, there is the difference of Sub-population GA with respect to single population GA that each sub-population of SPGA iterates in parallel and share each other their individuals that are called migrant to improve the solutions. There are many ways to implement SPGA but this study uses a different SPGA from the other SPGA. We use Euclidean Distance in migrating the solutions, thus we call it Euclidean Distance sub-population GA (EDSPGA). The Euclidean Distance will be explained in section of migration.

3. METHODOLOGY

3.1. General steps of genetic algorithm

SPGA begins with population initialization and being continued by crossover, mutation, evaluation, migration between sub-population and finally the selection. Based on the approaches in knapsack problems, we propose real-coded chromosome representation as shown in Figure 1. Each number represents the quantity of product stored.

The chromosomes which have been initialized are improved by a reproduction process consisting of one cut point crossover and random mutation. The crossover and mutation mechanisms are shown in Figure 2 and Figure 3. In the crossover process, each child inherits some of the genes from the parent. In the mutation process, some genes from the parent are shifted to produce a child. After reproduction process, then it goes into evaluation to combine the reproductive chromosomes with the existing population. The selected chromosomes will be passed to the next generation.

product 1	product 2	product 3	product 4	product 5	product 6	product 7	product 8	Fitness
4000	1500	2000	200	200	150	150	150	0.0009

Figure 1. Real-coded chromosome representation

				cutting point				
parent 1	40	300	270	26	193	340	810	162
parent 2	296	82	91	821	52	376	418	395
child 1	40	300	270	821	52	376	418	395
child 2	296	82	91	26	193	340	810	162

Figure 2. One cut poin crossover

		insertion point			selection point		
296	82	91	26	193	340	810	162
296	340	82	91	26	193	810	162
				shifting cells			

Figure 3. Insertion mutation

3.2. ED-Migration of SPGA

Each subpopulation improves separately. There is one individual in each subpopulation that will be migrated to the next subpopulation in order to improve the variety of solutions. The migrations scheme is presented in Figure 4.

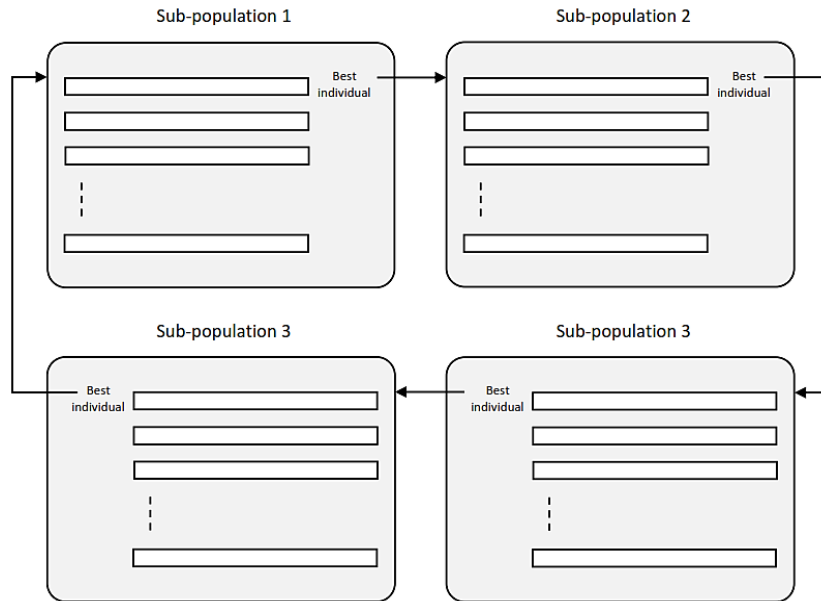


Figure 4. Migration scheme

Ten individuals from first subpopulation and the best individual from the next population have been chosen. The ten individuals of first subpopulation are compared to the best individual of next subpopulation by using one dimension Euclidean Distance. The Euclidean Distance formulation is presented in (1).

$$ED = \sqrt{(x_1 - x_2)^2} \quad (7)$$

ED means the distance between chromosome x_1 and x_2 . x_1 is a chromosome on the first sub-population and x_2 is a chromosome on the second sub-population. By applying Euclidean Distance formulation, we can find the chromosomes with the longest distance. The chromosome with the longest distance will replace the best individual in the next subpopulation.

4. EXPERIMENTAL RESULT AND ANALYSIS

This section explains the experimental results of all three methods performance consists of classical GA, SPGA and EDSPGA. The results refer to parameter testing of classical GA as the basis SPGA and EDSPGA parameter testing. First, we evaluate the parameter of classical GA that consist of population size (*popsize*) testing showed in Figure 5 for fitness value and Figure 6 for computational time. Figure 5 shows that the convergence point is at *popsize* 600. *Popsize* is tested from 10 and stopped at 1,000 because after *popsize* 600 there is no significant increase in fitness value. While it is showed in Figure 6 that the computational time is continuously increasing as the *popsize* value increases.

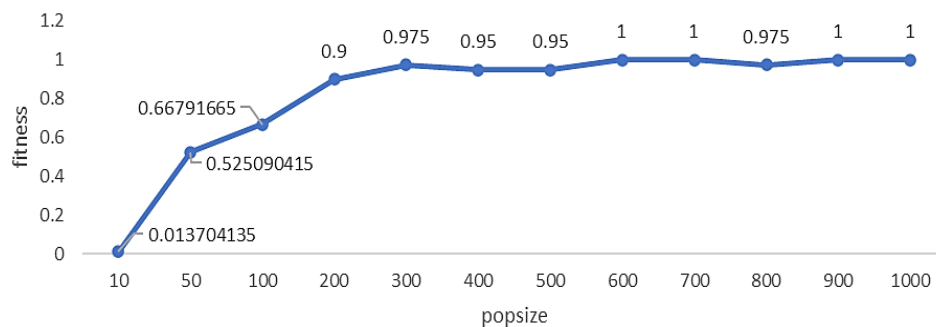


Figure 5. Fitness value in population size testing

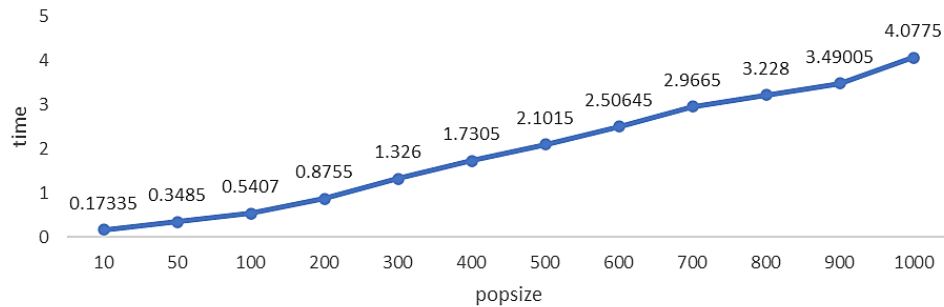


Figure 6. Computational time in population size testing

The next parameter test is number of generations testing showed in Figure 7 for fitness value and Figure 8 for computational time. Figure 7 shows that there is no significant change in the fitness value after 70 generations. Therefore, the best solution can be reached at 70 generations. Meanwhile, the computational time increase continuously as generation raises as shown in Figure 8. In this case, we found that small number of generations with the bigger population size can provide better result than small number of population size with bigger number of generations.

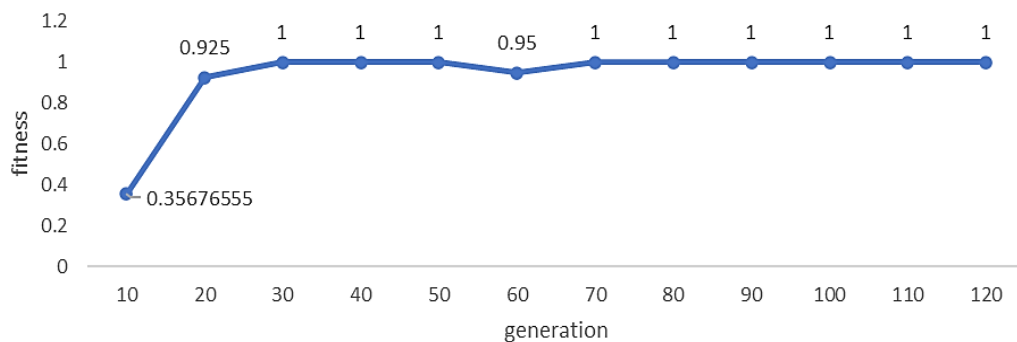


Figure 7. Fitness value in number of generations test

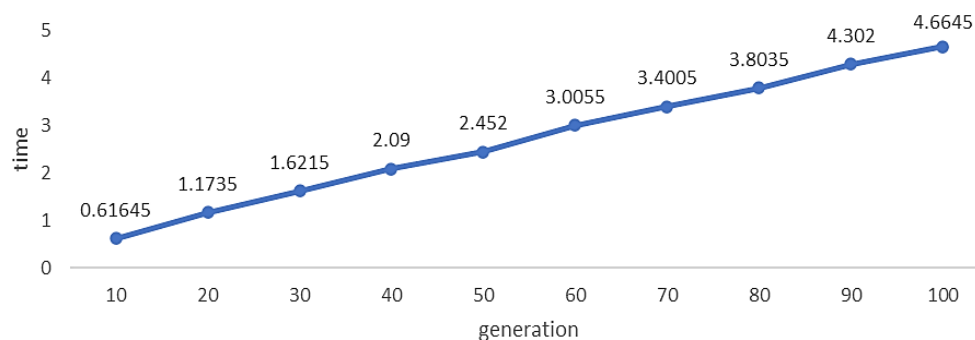
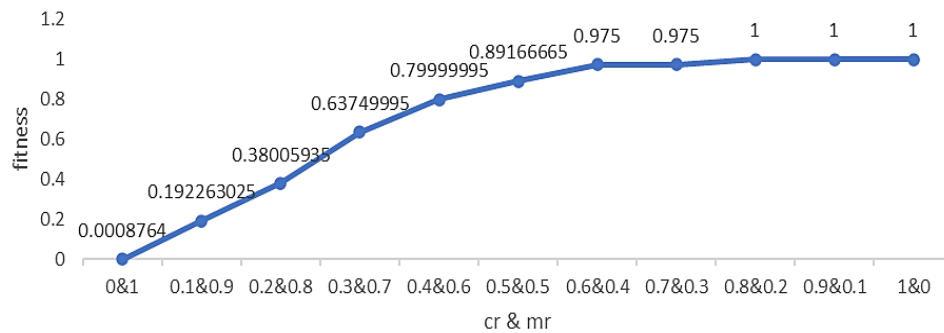
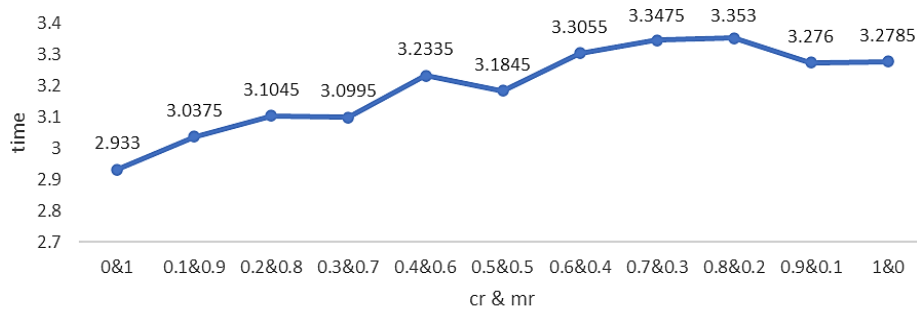


Figure 8. Computational time in number of generations test

Good result is a good solution that is provided not only with a good fitness value but also a short computational time. Therefore, we set the number of generations as 70 which is got from the test before with popsize 600. Last parameter testing of classical GA is crossover rate (*cr*) and mutation rate (*mr*) combination. The test uses 600 populations and 70 generations that is showed in Figure 9 for fitness value and Figure 10 for computational time.

Figure. 9 Fitness value in *cr* and *mr* testingFigure 10. Computational time in *cr* and *mr* testing

Test for *cr* and *mr* is carried out several times the best combination values of *cr* and *mr*. The values are used for population size and number of generation test. Figure 9 showed that the best value of *cr* and *mr* is 0.8 and 0.2 respectively. However, the computation time is lower in *cr* and *mr* of 0.9 and 0.1 with slightly lower fitness value. Therefore, *cr* and *mr* used in this case are 0.9 and 0.1. The best values of each parameter of classical GA is used to discover the best number of subpopulations in SPGA and EDSPGA that is showed in Figure 11 for fitness value and Figure 12 for computational time.

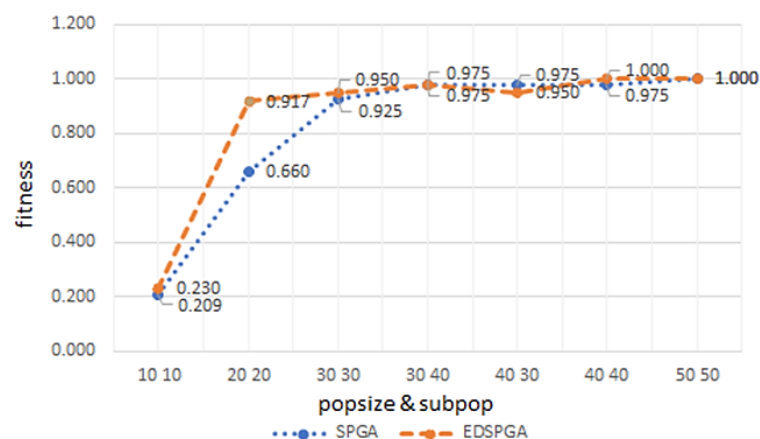


Figure. 11 Fitness comparison of SPGA and EDSPGA

Crossover and mutation rates for SPGA and EDSPGA are 0.9 and 0.1, referring to the results of the classical GA parameter tests that were carried out previously. The generation is determined to be 20 because there are no significant changes in the 20th generation of SPGA. Population size and number of sub-

population tests are carried out concurrently because high popsize does not always provide a better fitness value but provides a longer computational time referring to Figure 11. This is influenced by the generated number of sub-population. In Figure 8 it can be seen that population size of 40 and sub-population of 40 produce the highest fitness value and faster computation time than the larger number of popsize and subpop.

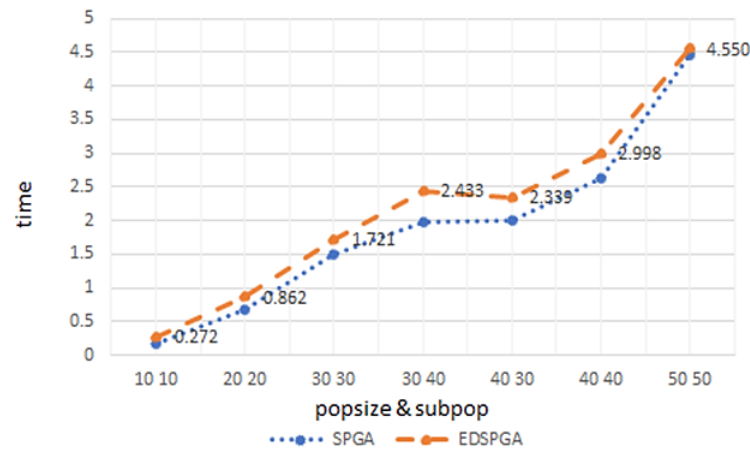


Figure 12. Computational time of SPGA and EDSPGA

Figure 11 shows that average fitness value of SPGA and EDSPGA are equally increase at population size of 30 and sub-population of 40 which is 0.975. Although at population size of 40 and sub-population of 30 EDSPGA decreased by 2.56% compared to SPGA, EDSPGA is able to provide a higher increase than SPGA without much different of the computational time from the previous population size of 20 and sub-population of 20. It shows that EDSPGA directly gives better fitness value not far from beginning. Therefore, we can say that the performance of EDSPGA is better than SPGA for the same number of popsizes and subpops. However, in this case, we found that in EDSPGA, a low populationsize along with a higher sub-population number provides better fitness value. This is indicated by an increase of fitness value on population size of 30 and sub-population number of 40 from 0.95 to 0.975 and a decrease in the fitness value on population-size 40 and sub-population number of 30 from 0.975 to 0.95. Based on the experiments that have been done, we compare the performance of SPGA with ED respect to classical GA and SPGA without ED. The comparison of fitness average is showed in Table 3.

Table 3. Test Result of GA, SPGA, and SPGA with eulidean distance

Test	GA		SPGA		EDSPGA	
	Fitness	Time	Fitness	Time	Fitness	Time
1	0.055555	0.158	1.0	2.81	1.0	2.1
2	0.142857	0.0624	1.0	2.62	1.0	2.17
3	0.058823	0.0625	1.0	2.56	1.0	1.04
4	0.076923	0.0625	1.0	2.63	1.0	2.08
5	0.166666	0.0817	1.0	2.51	1.0	2.04
6	0.333333	0.0625	0.5	2.67	1.0	2.1
7	0.25	0.0468	1.0	2.66	1.0	2.12
8	0.5	0.0781	1.0	2.45	1.0	2.06
9	0.333333	0.0683	1.0	2.71	1.0	2.02
10	0.02439	0.0575	1.0	2.77	1.0	2.13
11	0.071428	0.0625	1.0	2.57	1.0	2.24
12	0.05	0.0625	1.0	2.72	1.0	2.17
13	0.037037	0.0867	1.0	2.69	1.0	2.24
14	0.125	0.0539	1.0	2.53	1.0	2.16
15	0.5	0.0751	1.0	2.49	1.0	2.33
16	0.017543	0.0583	1.0	2.44	1.0	2.25
17	0.2	0.0697	1.0	2.62	1.0	2.27
18	0.166666	0.0625	1.0	2.76	1.0	2.1
19	0.111111	0.0708	1.0	2.84	1.0	2.27
20	0.25	0.0781	1.0	2.76	1.0	2.15
Average	0.173533	0.07102	0.975	2.6405	1.0	2.102

Table 3 summarizes the computational time of classical GA is 0.07102 seconds, SPGA is 2.6405 seconds and SPGA with Euclidean Distance is 2.012 seconds. SPGA reaches the higher fitness value of 2.102, while classical GA is 0.173533 and basic SPGA 0.95. Even the computational time of EDSPGA is longer than classical GA and basic SPGA, we have found that EDSPGA giving the best result in fitness average among classical GA and basic SPGA. Thus, it proves the effectiveness of the proposed migration mechanism to maintain population diversity and avoid an early convergence.

5. CONCLUSION

The computational experiment proves that GA, SPGA and EDSPGA could effectively solve the problem the knapsack problem. However, based on the same parameters with SPGA, GA only reaches 17.8% of SPGA fitness value. On the other hand, although EDSPGA requires a longer computational time, the result of EDSPGA is increased by 2.56% from SPGA fitness value. The next research as future work can be considered as i) discovering SPGA and EDSPGA performance by not only test popsize and subpop but also the generation and adaptive changing of crossover rate and mutation rate and ii) exploring the complexity of the problem and applying the method to solve more complex problems.




REFERENCES

- [1] L. Shen *et al.*, "Inventory optimization of fresh agricultural products supply chain based on agricultural superdocking," *J. Adv. Transp.*, pp. 1–13, Jan. 2020, doi: 10.1155/2020/2724164.
- [2] F. D. Croce, F. Salassa, and R. Scatamacchia, "An exact approach for the 0–1 knapsack problem with setups," *Comput. Oper. Res.*, vol. 80, pp. 61–67, Apr. 2017, doi: 10.1016/j.cor.2016.11.015.
- [3] Y. Feng and G.-G. Wang, "A binary moth search algorithm based on self-learning for multidimensional knapsack problems," *Futur. Gener. Comput. Syst.*, vol. 126, pp. 48–64, Jan. 2022, doi: 10.1016/j.future.2021.07.033.
- [4] Y. He and X. Wang, "Group theory-based optimization algorithm for solving knapsack problems," *Knowledge-Based Syst.*, vol. 219, May 2021, doi: 10.1016/j.knosys.2018.07.045.
- [5] I. M. Ali, D. Essam, and K. Kasmarik, "Novel binary differential evolution algorithm for knapsack problems," *Inf. Sci. (Ny)*, vol. 542, pp. 177–194, Jan. 2021, doi: 10.1016/j.ins.2020.07.013.
- [6] N. Thongsri, P. Warintarawej, S. Chotkaew, and W. Saetang, "Implementation of a personalized food recommendation system based on collaborative filtering and knapsack method," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 1, pp. 630–638, Feb. 2022, doi: 10.11591/ijece.v12i1.pp630-638.
- [7] C. Changdar, G. S. Mahapatra, and R. K. Pal, "An improved genetic algorithm based approach to solve constrained knapsack problem in fuzzy environment," *Expert Syst. Appl.*, vol. 42, no. 4, pp. 2276–2286, Mar. 2015, doi: 10.1016/j.eswa.2014.09.006.
- [8] Y. Zhou, M. Zhao, M. Fan, Y. Wang, and J. Wang, "An efficient local search for large-scale set-union knapsack problem," *Data Technol. Appl.*, vol. 55, no. 2, pp. 233–250, Apr. 2021, doi: 10.1108/DTA-05-2020-0120.
- [9] Q. Zhai, Y. He, G. Wang, and X. Hao, "A general approach to solving hardware and software partitioning problem based on evolutionary algorithms," *Adv. Eng. Softw.*, vol. 159, Sep. 2021, doi: 10.1016/j.advengsoft.2021.102998.
- [10] X. Yang, Y. Zhou, A. Shen, J. Lin, and Y. Zhong, "A hybrid ant colony optimization algorithm for the knapsack problem with a single continuous variable," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Jun. 2021, pp. 57–65, doi: 10.1145/3449639.3459343.
- [11] O. Kabadurmus, M. F. Tasgetiren, H. Oztop, and M. S. Erdogan, "Solving 0-1 Bi-objective multi-dimensional knapsack problems using binary genetic algorithm," in *Studies in Computational Intelligence*, vol. 906, 2021, pp. 51–67, doi: 10.1007/978-3-030-58930-1_4.
- [12] A. Syarif, D. Anggraini, K. Muludi, W. Wamiliana, and M. Gen, "Comparing various genetic algorithm approaches for multiple-choice multi-dimensional knapsack problem (mm-KP)," *Int. J. Intell. Eng. Syst.*, vol. 13, no. 5, pp. 455–462, Oct. 2020, doi: 10.22266/ijies2020.1031.40.
- [13] A. Rahmi, W. F. Mahmudy, and M. Z. Sarwani, "Genetic algorithms for optimization of multi-level product distribution," *Int. J. Artif. Intell.*, vol. 18, no. 1, pp. 135–147, 2020.
- [14] Q. Kotimah, W. F. Mahmudy, and V. N. Wijayaningrum, "Optimization of fuzzy Tsukamoto membership function using genetic algorithm to determine the river water," *Int. J. Electr. Comput. Eng.*, vol. 7, no. 5, pp. 2838–2846, Oct. 2017, doi: 10.11591/ijece.v7i5.pp2838-2846.
- [15] V. Meilia, B. D. Setiawan, and N. Santoso, "Extreme learning machine weights optimization using genetic algorithm in electrical load forecasting," *J. Inf. Technol. Comput. Sci.*, vol. 3, no. 1, pp. 77–87, 2018.
- [16] Z. A. Ali, S. A. Rasheed, and N. N. Ali, "An enhanced hybrid genetic algorithm for solving traveling salesman problem," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 18, no. 2, pp. 1035–1039, 2020, doi: 10.11591/ijeecs.v18.i2.pp1035-1039.
- [17] S. D L, "Energy efficient intelligent routing in WSN using dominant genetic algorithm," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 1, pp. 500–511, Feb. 2020, doi: 10.11591/ijece.v10i1.pp500-511.
- [18] A. M. Hemeida, O. M. Bakry, A.-A. A. Mohamed, and E. A. Mahmoud, "Genetic algorithms and satin bowerbird optimization for optimal allocation of distributed generators in radial system," *Appl. Soft Comput.*, vol. 111, Nov. 2021, doi: 10.1016/j.asoc.2021.107727.
- [19] A. El Beqal, B. Benhala, and I. Zorkani, "A genetic algorithm for the optimal design of a multistage amplifier," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 1, pp. 129–138, Feb. 2020, doi: 10.11591/ijece.v10i1.pp129-138.
- [20] W. F. Mahmudy, M. Z. Sarwani, A. Rahmi, and A. W. Widodo, "Optimization of multi-stage distribution process using improved genetic algorithm," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 2, pp. 211–219, 2021.
- [21] K. Kamil, K. H. Chong, H. Hashim, and S. A. Shaaya, "A multiple mitosis genetic algorithm," *IAES Int. J. Artif. Intell.*, vol. 8, no. 3, pp. 252–258, Dec. 2019, doi: 10.11591/ijai.v8.i3.pp252-258.
- [22] S. Malik and S. Wadhwa, "Preventing premature convergence in genetic algorithm using DGCA and elitist technique," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 4, no. 6, pp. 410–418, 2014.




- [23] S. Masrom, M. Mohamad, S. M. Hatim, N. Baharun, N. Omar, and A. S. Abd. Rahman, "Different mutation and crossover set of genetic programming in an automated machine learning," *IAES Int. J. Artif. Intell.*, vol. 9, no. 3, pp. 402–408, Sep. 2020, doi: 10.11591/ijai.v9.i3.pp402-408.
- [24] G. E. Yulastuti, A. Mustika, W. Firdaus, and I. Pambudi, "Optimization of multi-product aggregate production planning using hybrid simulated annealing and adaptive genetic algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 11, pp. 484–489, 2019, doi: 10.14569/IJACSA.2019.0101167.
- [25] A. P. Rifai, P. A. Kusumastuti, S. T. W. Mara, R. Norcahyo, and S. Z. Md Dawal, "Multi-operator hybrid genetic algorithm-simulated annealing for reentrant permutation flow-shop scheduling," *ASEAN Eng. J.*, vol. 11, no. 3, pp. 109–126, Apr. 2021, doi: 10.11113/aej.v11.16875.
- [26] A. Iranmanesh and H. R. Naji, "DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing," *Cluster Comput.*, vol. 24, no. 2, pp. 667–681, Jun. 2021, doi: 10.1007/s10586-020-03145-8.
- [27] A. K. Ariyani, W. F. Mahmudy, and Y. P. Anggodo, "Hybrid genetic algorithms and simulated annealing for multi-trip vehicle routing problem with time windows," *Int. J. Electr. Comput. Eng.*, vol. 8, no. 6, pp. 4713–4723, 2018, doi: 10.11591/ijece.v8i6.pp.4713-4723.
- [28] A. A. K. Taher and S. M. Kadhim, "Improvement of genetic algorithm using artificial bee colony," *Bull. Electr. Eng. Informatics*, vol. 9, no. 5, pp. 2125–2133, Oct. 2020, doi: 10.11591/eei.v9i5.2233.
- [29] M. L. Seisarrina, I. Cholisodin, and H. Nurwarsito, "Invigilator examination scheduling using partial random injection and adaptive time variant genetic algorithm," *J. Inf. Technol. Comput. Sci.*, vol. 3, no. 2, pp. 113–119, Nov. 2018, doi: 10.25126/jitecs.20183250.
- [30] F. Uysal, R. Sonmez, and S. K. Isleyen, "A graphical processing unit-based parallel hybrid genetic algorithm for resource-constrained multi-project scheduling problem," *Concurr. Comput. Pract. Exp.*, vol. 33, no. 16, Aug. 2021, doi: 10.1002/cpe.6266.
- [31] A. Marrero, E. Segredo, and C. Leon, "A parallel genetic algorithm to speed up the resolution of the algorithm selection problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Jul. 2021, pp. 1978–1981, doi: 10.1145/3449726.3463160.
- [32] W. N. Abdullah and S. A. Alagha, "A parallel adaptive genetic algorithm for job shop scheduling problem," *J. Phys. Conf. Ser.*, vol. 1879, no. 2, May 2021, doi: 10.1088/1742-6596/1879/2/022078.
- [33] X. Shi, W. Long, Y. Li, and D. Deng, "Multi-population genetic algorithm with ER network for solving flexible job shop scheduling problems," *PLoS One*, vol. 15, no. 5, May 2020, doi: 10.1371/journal.pone.0233759.

BIOGRAPHIES OF AUTHORS






Wayan Firdaus Mahmudy    obtained a Bachelor of Science degree from the Mathematics Department, Brawijaya University in 1995. His Master in Informatics Engineering degree was obtained from the Sepuluh Nopember Institute of Technology, Surabaya in 1999 while a Ph.D. in Manufacturing Engineering was obtained from the University of South Australia in 2014. He is a Professor at Department of Computer Science, Brawijaya University (UB), Indonesia. His research interests include optimization of combinatorial problems and machine learning. He can be contacted at email: wayanfm@ub.ac.id.



Nindynar Rikatsih    received Bachelor of Computer degree and her Master in Computer Science degree from Faculty of Computer Science, Brawijaya University (UB) in 2016 and 2020 respectively. She is currently a lecturer in Institut Teknologi, Science and Health of Hospital dr. Soepraosen, Malang. Her research interests include evolutionary algorithms, optimization problems, and data mining. She can be contacted at email: R.Nindynar@gmail.com.



Syafrial    received Bachelor of Socio-economic degree from Faculty of Agriculture, Brawijaya University (UB) in 1982 and received Master and Doctor degree in the field of Agricultural Economic Science, Bogor Agricultural University in 1986 and 2003 respectively. He is currently a senior lecturer of Agricultural Socio-Economic Department in Faculty of Agriculture, Universitas Brawijaya. He can be contacted at email: syafrial.fp@ub.ac.id.